

大会報告書

所属(電気通信大学高玉研究室)
Cake's

内容

- 1 チームについて
- 2 CanSat 機体概要
- 3 大会結果
 - 3.1 能代宇宙イベント
 - 3.2 ARLISS
- 4 まとめ

2016 年 10 月 12 日
電気通信大学高玉研究室

1 チームについて

指導教員 高玉圭樹

学年	名前	製作担当箇所
M1	松本和馬	リーダー/ソフトウェア(班長)/会計
M1	梅内祐太	プロジェクトマネージャー
B4	上原知里	ハードウェア/デザイナー
M2	辰巳嵩豊	ソフトウェア/ロジ
B2	張翌坤	ハードウェア
B3	永間慎太郎	回路
B4	則武和輝	ハードウェア(班長)
B4	福田千賀	ハードウェア/デザイナー/ロジ
B4	村上大和	回路(班長)/ソフトウェア
M2	村田暁紀	ソフトウェア/広報/メンター

我が研究室が CanSat を作成する理由は、チーム開発を通して協調性と技術力を上げるためである。最終的なゴールは ARLISS でのミッションの成功であるが、その予行練習の場として、能代宇宙イベントにも参加した。

2 CanSat 機体概要

2.1 ミッションステートメント

パラシュートで着地した CanSat がそれを切り離し、自律制御で予め定められたゴール地点への走行を目標とする。

CanSat は従来の二輪ローバータイプではなく、一輪クローラータイプとする。一輪クローラータイプは接地面積が二輪ローバータイプと比べて大きいため、スタックしにくい性質があり、その有効性を検証する。

2.2 サクセスクライテリア

ARLISS におけるサクセスクライテリアは次の表の通りである。

表 2.2.1: ARLISS サクセスクライテリア

	内容
ミニマムサクセス	・ 100m 以上走行することができる。
フルサクセス	・ 現地の走行実験において従来の二輪ローバーに比べ、50%少ないスタック回数で砂地を走行する。 ・ ゴール地点から 10m 以内で停止する。
アドバンスドサクセス	・ CanSat が超えられない轍によって進行不可能な時に脱出機構及び脱出プログラムによって昨年度製作した CanSat より 50%早く脱出することができる。 ・ センサーの値により、左右のモーターのレシオを自動調整し、直進制御することができる。

能代宇宙イベントでのサクセスクライテリアは次の表の通りである。

表 2.2.2: 能代宇宙イベントでのサクセスクライテリア

	内容
ミニマムサクセス	・ パラシュートを切り離すことができる. ・ 100m 以上走行することができる.
ミドルサクセス	・ 方向転換において要求された角度に対する誤差を 15 度以内に抑えることができる.
フルサクセス	・ 二輪ローバーがスタックする轍や叢にスタックせずに走行できる. ・ ゴール地点から 10m 以内に停止する.
アドバンスドサクセス	・ CanSat が通常走行で越えられない轍や叢によって進行不可能な時に脱出機構及び脱出プログラムによって脱出できる.

2.3 ミッションシーケンス

キャリア収納からミッション達成までの流れは次のとおりである.

1. キャリアに CanSat を収納する
2. ロケットから CanSat を降下させる
3. CanSat のパラシュートが開傘する
4.
 - a. 地上に着地する
 - b. パラシュートを切り離す
5.
 - a. ゴールに向かって走行する
 - b. 轍にスタックしたら, 脱出機構で脱出する
6.
 - a. ゴール付近は減速する
 - b. ゴール判定し, 停止する

これを図 2.3.1 に図示した.

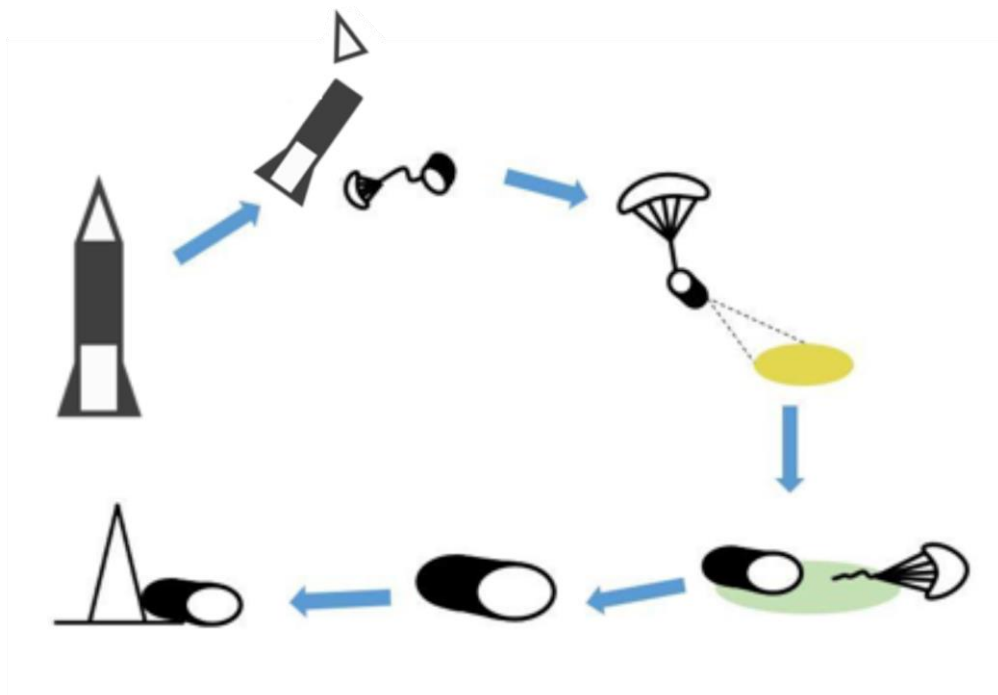


図 2.3.1: ミッションシーケンスの流れ

2.4 製作スケジュール

チーム内ミーティングの頻度・・・週1回

※全て 2016 年

【チーム全体】

- チーム発足・規約・フォーマットなどの作成: 4 月
- 「轍にハマらない」コンセプト決定: 4 月
- スマートフォンを使用することに決定: 4 月
- 形状の検討・「クローラー型」に決定: 5 月
- 無線通信規格の検討: 6 月
- パラシュートの選定: 6 月
- モーターの選定: 6 月
- パラシュート切り離し機構の検討: 6-7 月
- 履帯の素材の選定・実験: 5-8 月
- 方向転換メソッドの開発: 5-8 月
- ソフトウェア開発開始: 7 月
- 第1回安全審査に向けた各種試験: 7 月
- 初号機完成: 7 月
- 合同気球試験: 8 月
- 第2回安全審査に向けた各種試験: 8 月
- 式号機完成: 8 月
- チェックリストの作成: 8 月
- 能代宇宙イベント: 8 月
- ブラッシュアップ: 8-9 月
- 参号機完成: 8 月
- ARLISS: 9 月

【ソフトウェア】

- マイコン(スマートフォン)の選定: 4 月
- 通信プロトコルの決定・検証: 5-6 月
- Wi-Fi 通信プログラム: 6-7 月
- コントローラープログラム: 6-8 月
- アルゴリズム・クラス設計: 6 月
- 開発環境構築: 7 月
- センサー値の取得: 7 月
- ロガー: 7 月
- シーケンス: 7-8 月
- 旋回メソッド/Navigation アルゴリズムの改良: 8-9 月
- 設定画面: 8 月
- 耐熱対策: 9 月

【ハードウェア】

- コンセプトを達成するためのアイデアに対する有効性実験：4 月
- BBM 設計開始：5 月
 - 形状，素材の検討
- BBM 検証終了：6 月上旬
- EM 設計開始：6 月中旬
 - 履帯素材の検討開始：6 月上旬
 - パラシュート分離機構の設計：6 月上旬
 - スタビライザーの設計：6 月中旬
 - シャーシの設計：6 月下旬スタート
 - 統合設計：7 月上旬
- EM 設計終了：7 月中旬
- EM 試験開始：7 月中旬
- 履帯素材決定：7 月下旬
- EM 試験終了：8 月上旬
- FM1 号機(名：ベイマックス)量産開始：8 月上旬
- FM1 号機作成終了：能代イベント前
- FM1 号機試験開始：8 月中旬
- FM2 号機(名：BB8)量産開始：8 月下旬
- FM2 号機量産終了：8 月下旬
- 予備物品作成：9 月上旬

【回路】

- Wi-Fi モジュール，モータードライバの選定：4 月
- モータードライバの検証：5-6 月
- マイコンプログラムの作成開始：6 月
- 回路基板(名：ショート)作成：6 月
- モーターからの音楽を奏でる：7 月
- 通信が途切れたときの対策：8 月
- 回路基板(名：パイ)作成：8 月
- 回路基板(名：タルト)作成：8 月
- deep sleep 機能の実装：8 月
- deep sleep 機能による回路の変更：8 月
- 耐熱対策：9 月
- 命令間引き対策：9 月

2.5 システム図

システムは図 2.5.1 のようになっている.

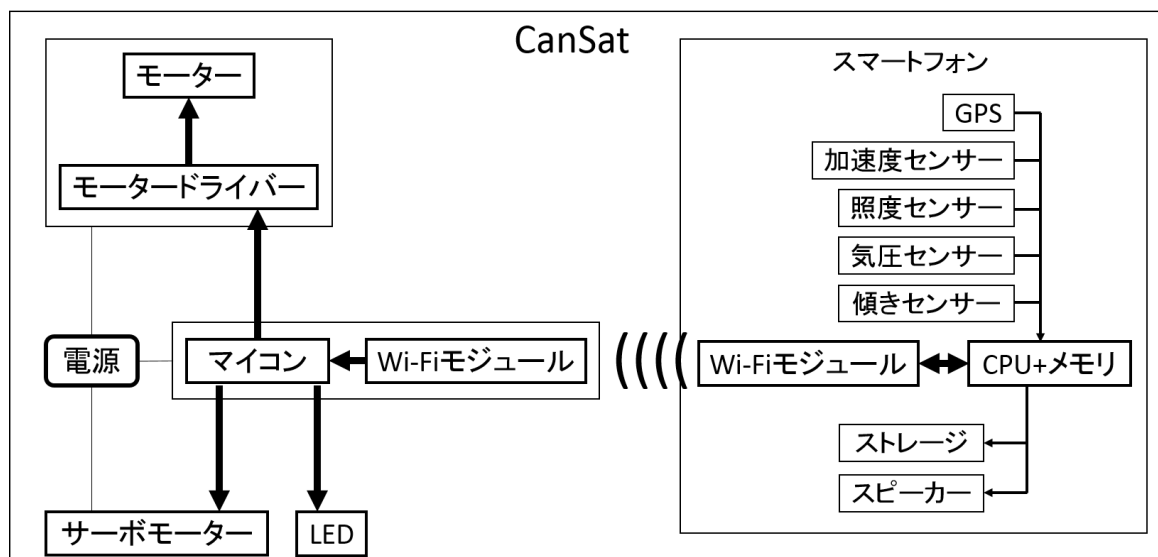
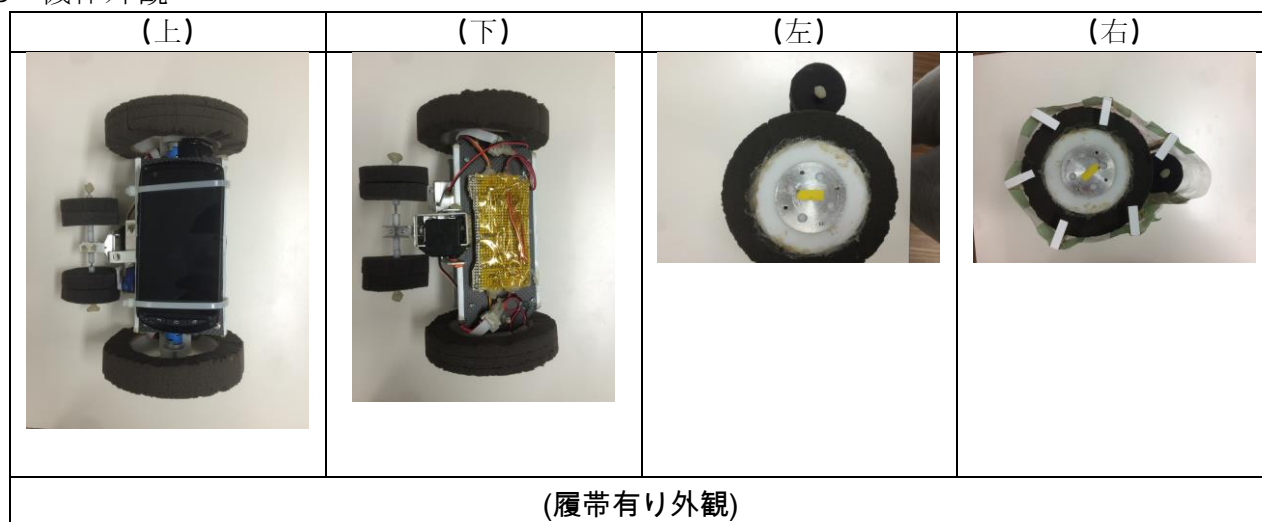


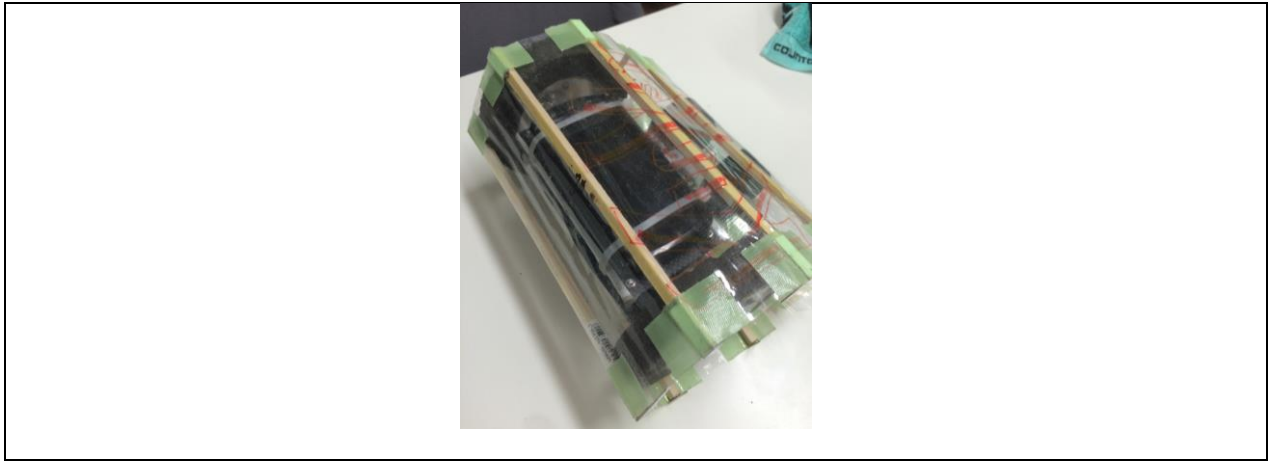
図 2.5.1: システム図

センサー及び演算はスマートフォンを用いる. Wi-Fi モジュールを搭載したマイコンがスマートフォンが出した司令を基に実際にモーターやサーボモーターなどの出力を行う. スマートフォンとマイコンを有線の通信ではなく, 無線を用いているのは接続コネクタの破損により通信遮断を防ぐため及びモジュールを完全に分離することで回路の破損やショートなどの影響を他のモジュールに与えないためである.

この CanSat の使用する無線規格は Wi-Fi(IEEE 802.11n)であり, 使用する機器の型番は ESP8266(モジュール)及び SKT01(スマートフォン)である.

2.6 機体外観





2.7 機体構造・仕組み

以下、クローラー型である自団体の CanSat の周りを覆っている塩化ビニルシートを履帯と呼ぶ

- 基本構造

図 2.7.1 と図 2.7.2 に履帯なしで前方と後方から見た機体を示す。重心の安定を図るため、動力軸であるモーターを中心に配置した。上側には GPS を取得するスマートフォンを配置し、回路はモーターをかえして反対側の機体下部に取り付けた。また、電源は単三電池を使用しており、スマートフォンの下に位置している。モーター付いている 1 段目のシャーシとスマートフォンが乗っている 2 段目のシャーシがある。1 段目のシャーシにすべての機構がつながっている。図 2.7.1 と図 2.7.2 写真からもわかるように、1 段目のシャーシにはモーターホルダー、スタビライザー & パラ切り離し機構、スマホ固定台、回路がついている。1 段目と 2 段目はジュラコンスペーサーと中にネジを通した支柱で繋がっている。以下、各機構について説明を行う。

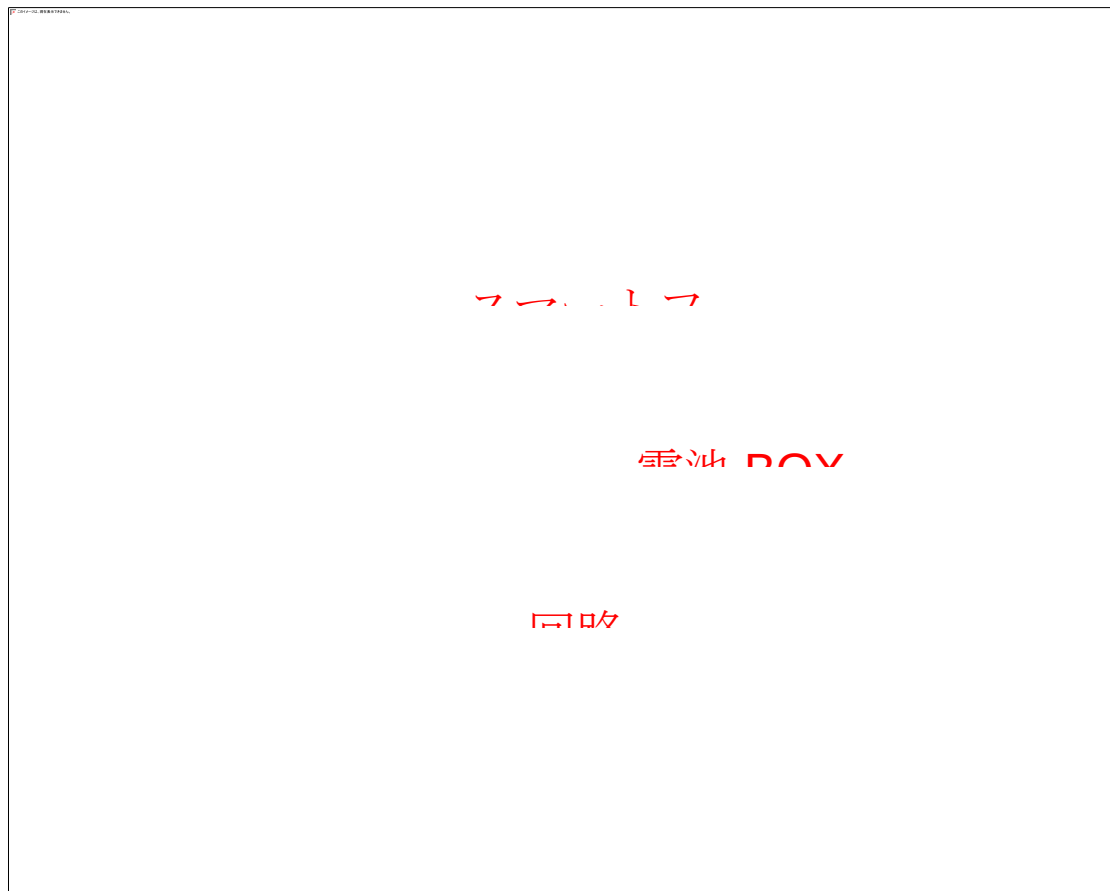


図 2.7.1: 機体のみ，前面

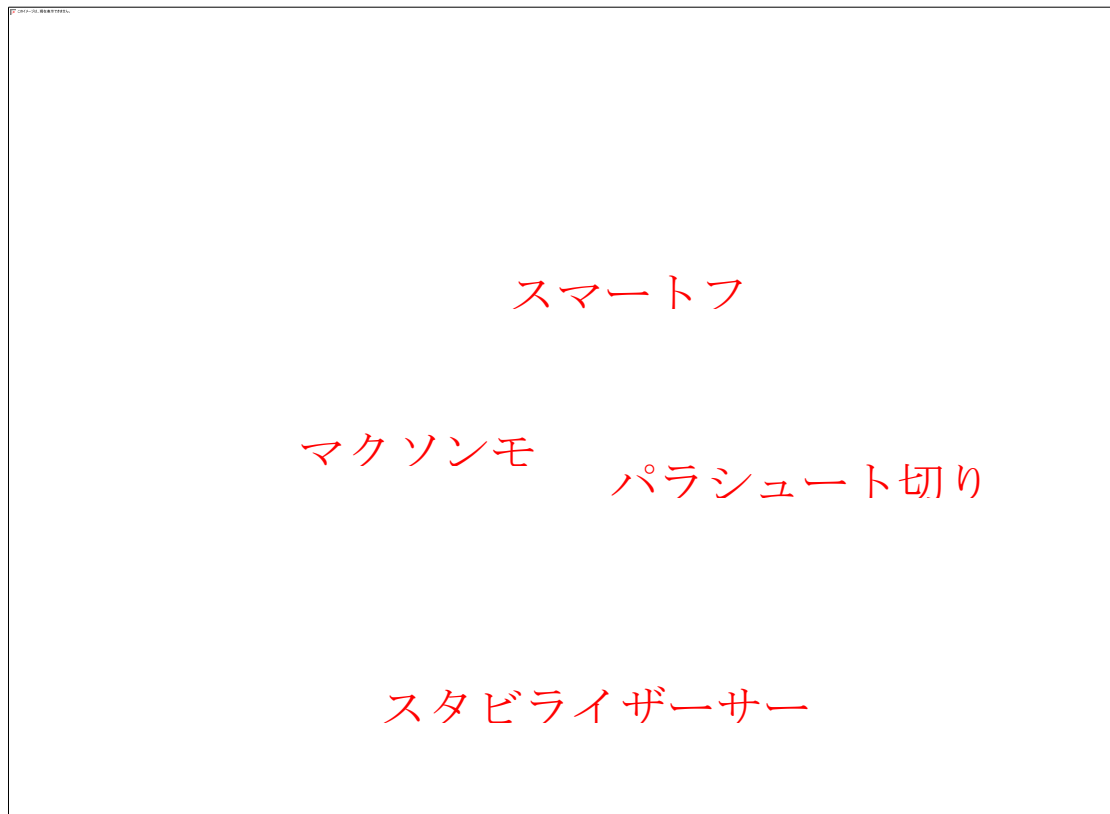


図 2.7.1: 機体のみ，後面

- 履帯構造

図 2.7.3 (左) は履帯を内側から，図 2.7.4 (右) は履帯を横から見た図である。主に塩化ビニルでできている履帯で機体 (図 2.7.1) を包むことで，スタックしにくく走破性に優れた CanSat を実現した。履帯の内側には，①塩化ビニルのシートがたわまないように，②機体 (図 2.7.1) に取り付けられているタイヤが履帯内で空回りしないように，ロットを 6 本貼り付けている。また，ロットの両端には機体 (図 2.7.1) がはみ出てしまわないように横抑えを図 2.7.4 (右) のように取り付けている。さらに，パラシュート切り離し機構にパラシュートを留めるための穴が 2 つ空いている。これは，以下で説明する切り離しピンとパラ紐を通すために設けられたものである。

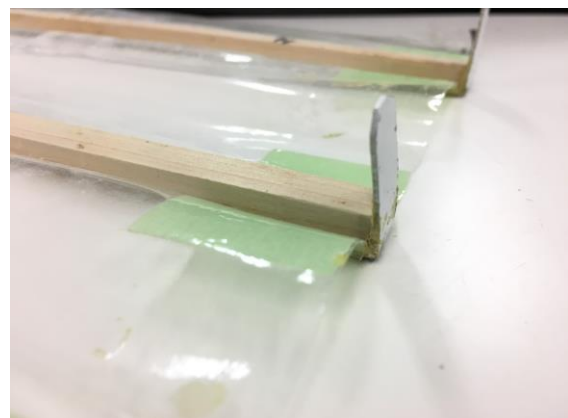


図 2.7.3: 履帯構造

- パラシュート

図 2.7.4 にパラシュートの概観を示す。パラ紐によって”パラシュート”，”絡まり防止機構”，”切り離しピン”が繋がっている。絡まり防止機構はメジャーを 2 つ”() ”のような形で張り合わせたものである。この機構は，展開によってパラシュートと切り離しピンを遠ざけることによって CanSat とパラシュートが絡まってしまうことを防止する。切り離しピンを図 2.7.5 に示す。ジュ

ラルミンを旋盤によって加工して作られたもので、釘を通す穴が開いている。また、切り離しの際に遠くへ離すことを目的としてバネが取り付けられている。

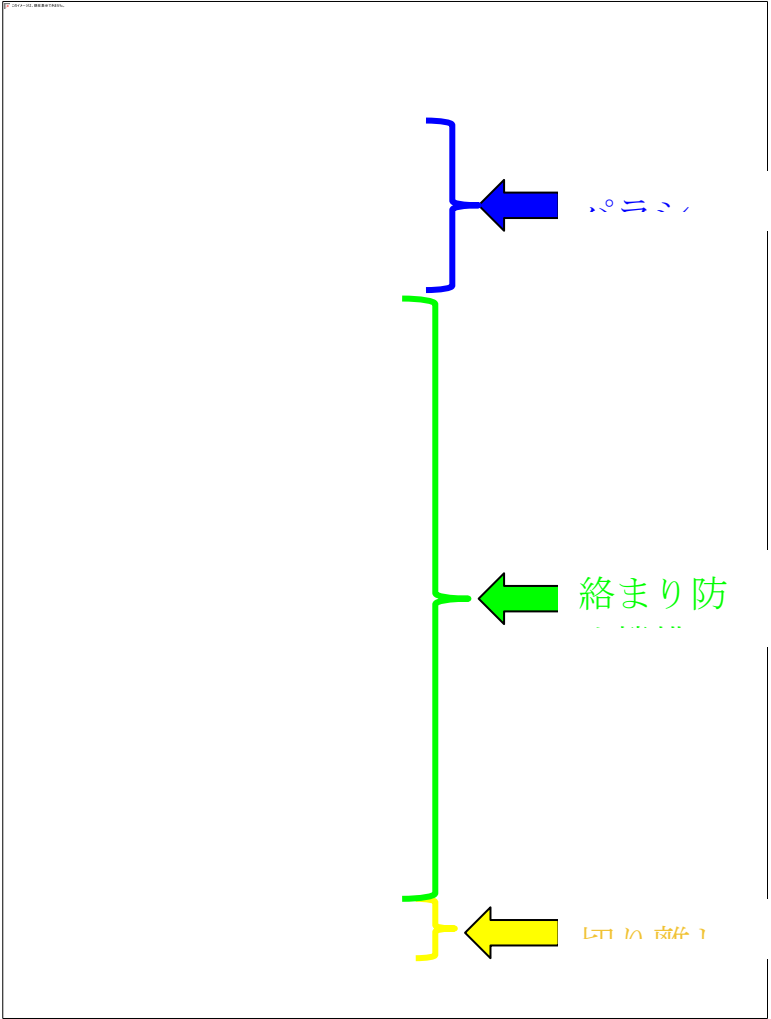


図 2.7.4: パラシュート概観

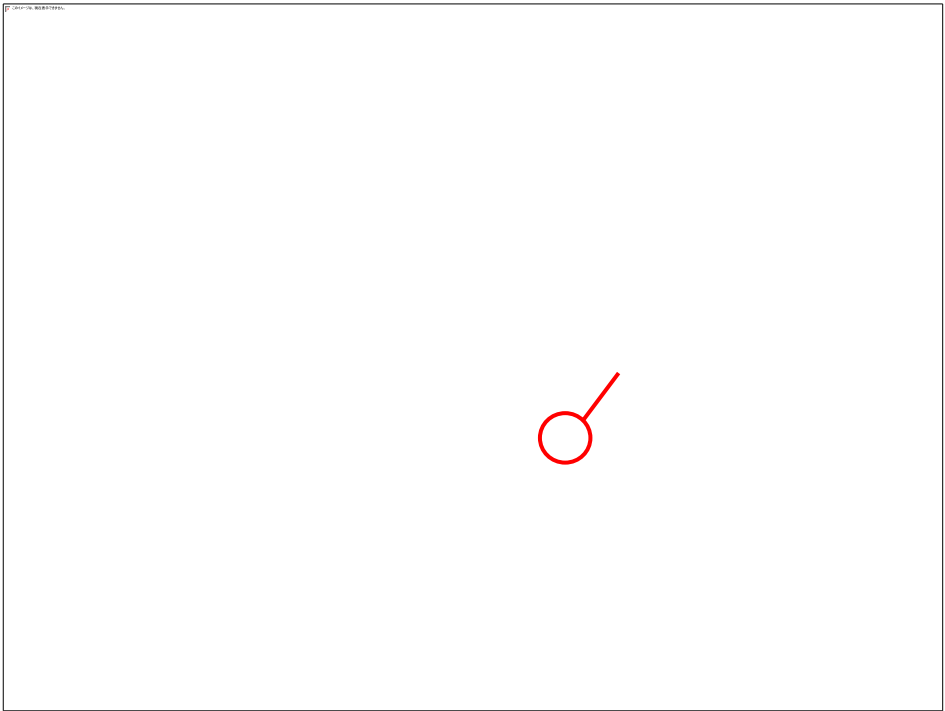


図 2.7.5: 切り離しピン

- パラシュート切り離し機構

図 2.7.6 にパラシュート切り離し機構を示す。パラシュートについている切り離しピンを、パラシュート切り離し機構に通し、さらに切り離しピンに開いている穴に釘を通して固定を行う。切り離しの際は、パラ切り離しサーボモーターを動作させ、釘を引き抜くことで切り離しピンが切り離される。また、パラ紐の輪っかを通して理由は、力の働く方向をバネの伸縮方向と異なる方向にすることでバネが縮み、釘が外れないようにするためである。さらに、パラ切り離し機構に働く力も垂直にすることで、アルミが曲がってしまうことを防止している。実際の動きは以下の動画で確認できる。

<https://youtu.be/uCDPOgPssvw>

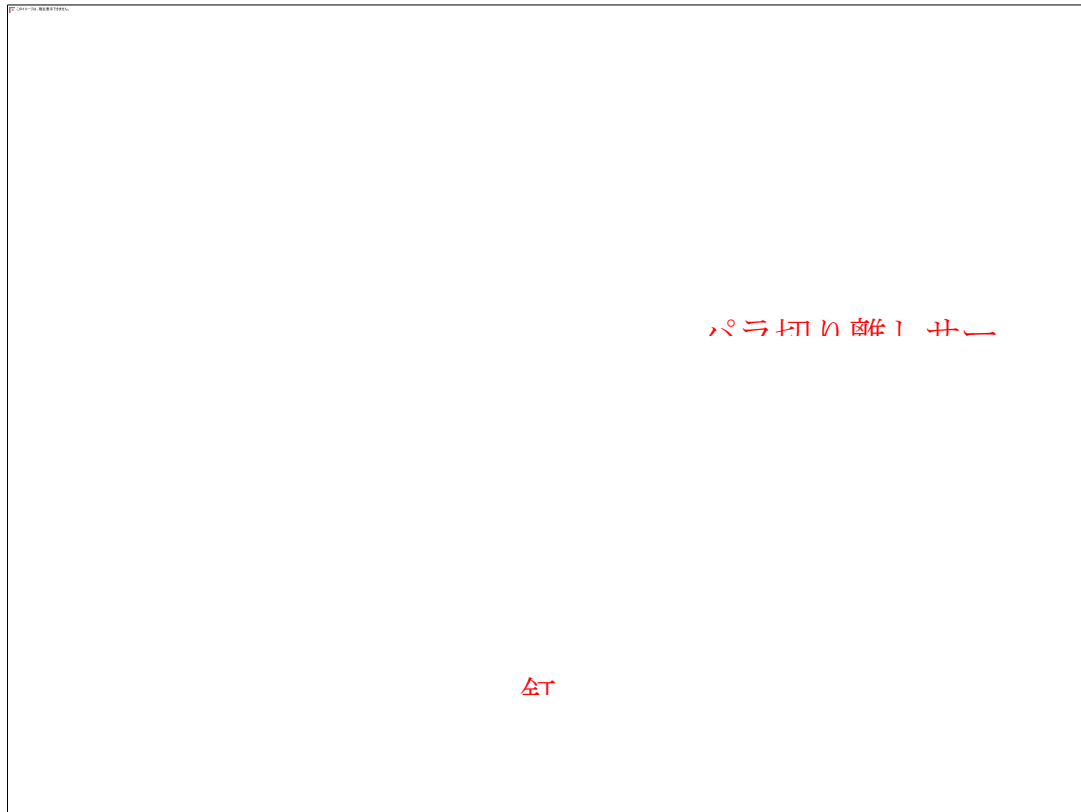


図 2.7.6: パラシュート切り離し機構

- スタビライザー

図 2.7.7 にスタビライザーの概観を示す。ブラケットはシャーシとサーボを繋げる役目と、スタビ爪(下図金属 T 字部分)を付ける目的がある。そのスタビ爪は後輪コネクタをかえして後輪と繋がっている。キャリア収納時は図 2.7.9 の左図のようになっていて、走行時は右図のようになっている。また、轍にはまった際は前後に動かすことによって脱出行動を行う。

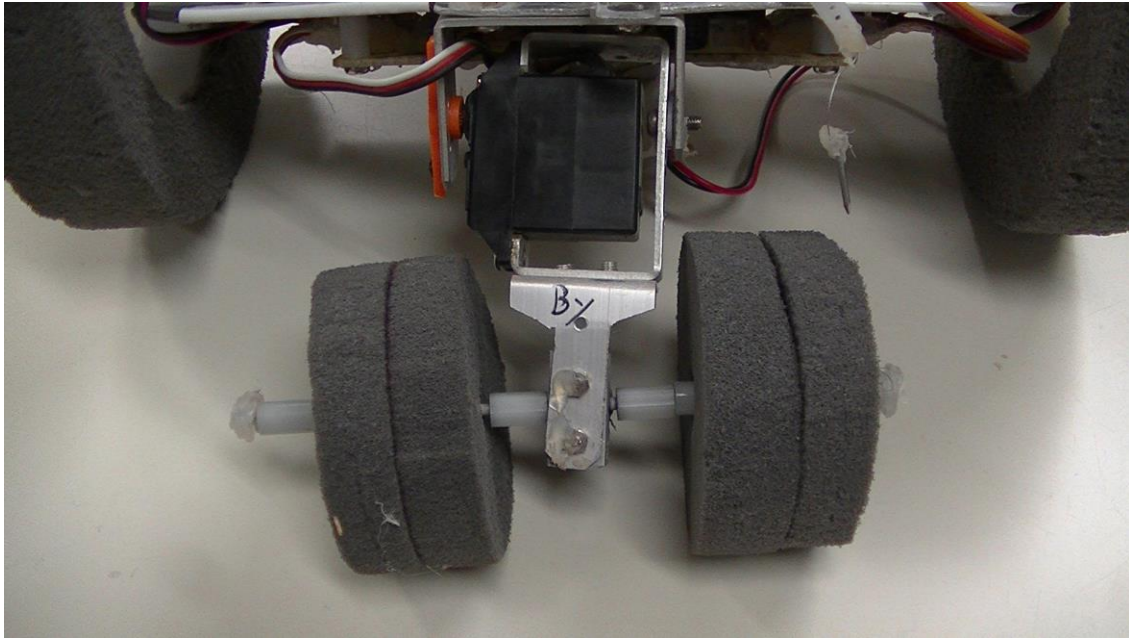


図 2.7.7: スタビライザー概観

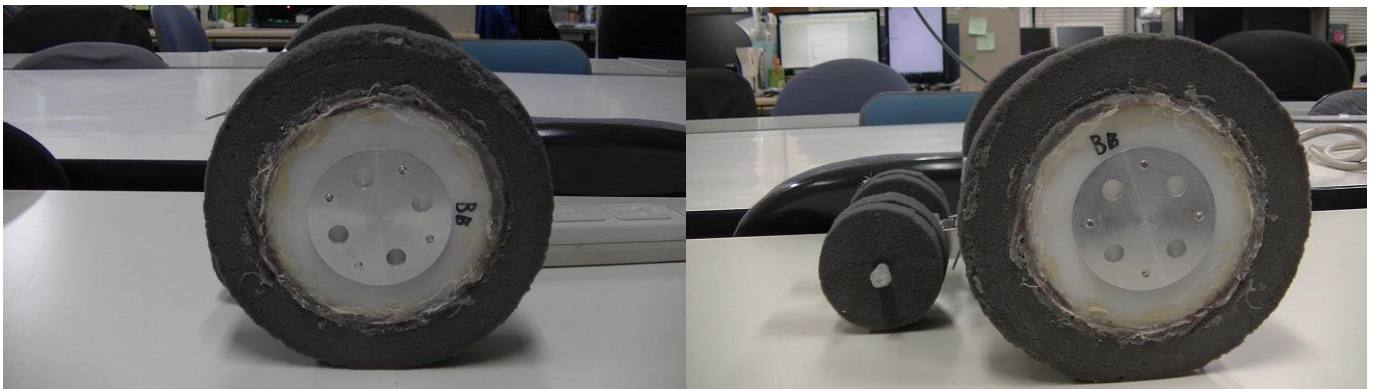


図 2.7.9: スタビライザー 展開後(左) 展開後(右)

- モーター周り

モーター周りの概観を図 2.7.10 に示す。モーターホルダーがモーターをホールドしシャーシと繋がっている。また動力は、モーター→カップリング→(ベアリング)→車軸→タイヤ→履帯のように伝わっている。モーターをカップリングで車軸と繋ぐことで出力軸が曲がらない対策を施している。また、モーターホルダーでしっかり固定することによって破損などの対策をしている。さらに、モーターホルダーはカップリングをかえて車軸の半径方向も固定しているため安定した走りを保ちながらも故障の対策をしている。(図 2.7.11, 12 参照)

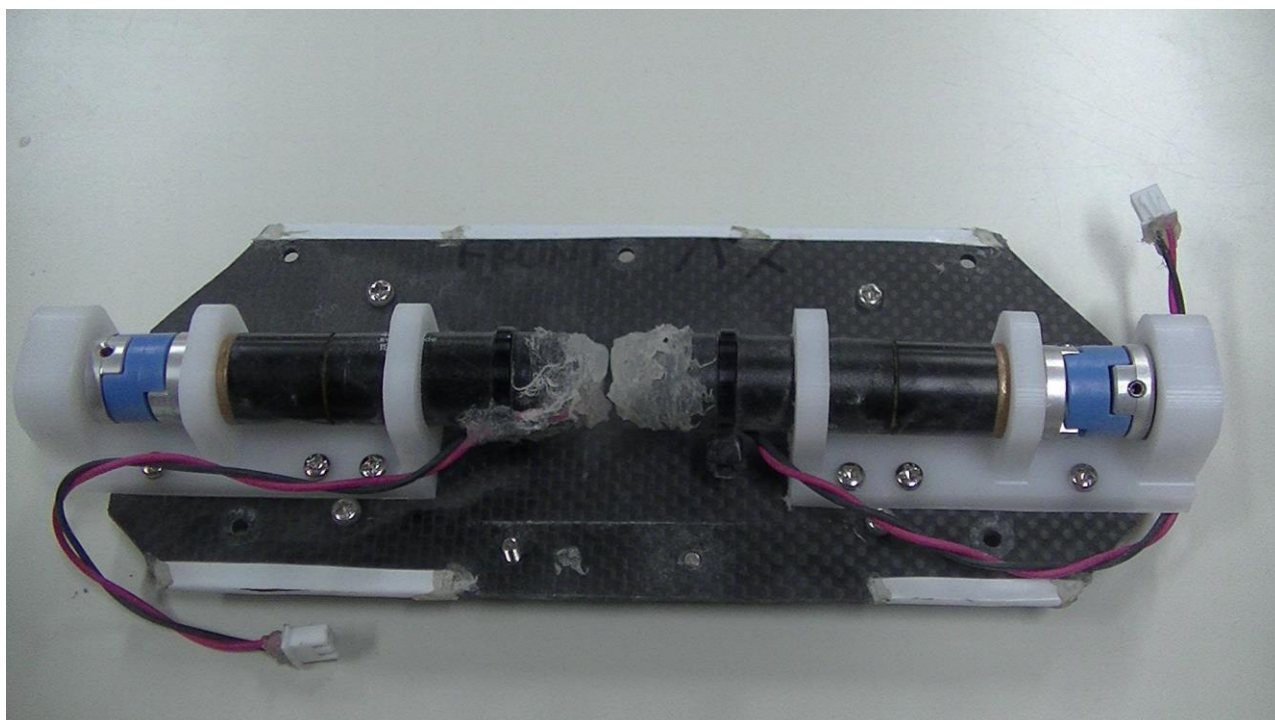


図 2.7.10: モーター周り概観

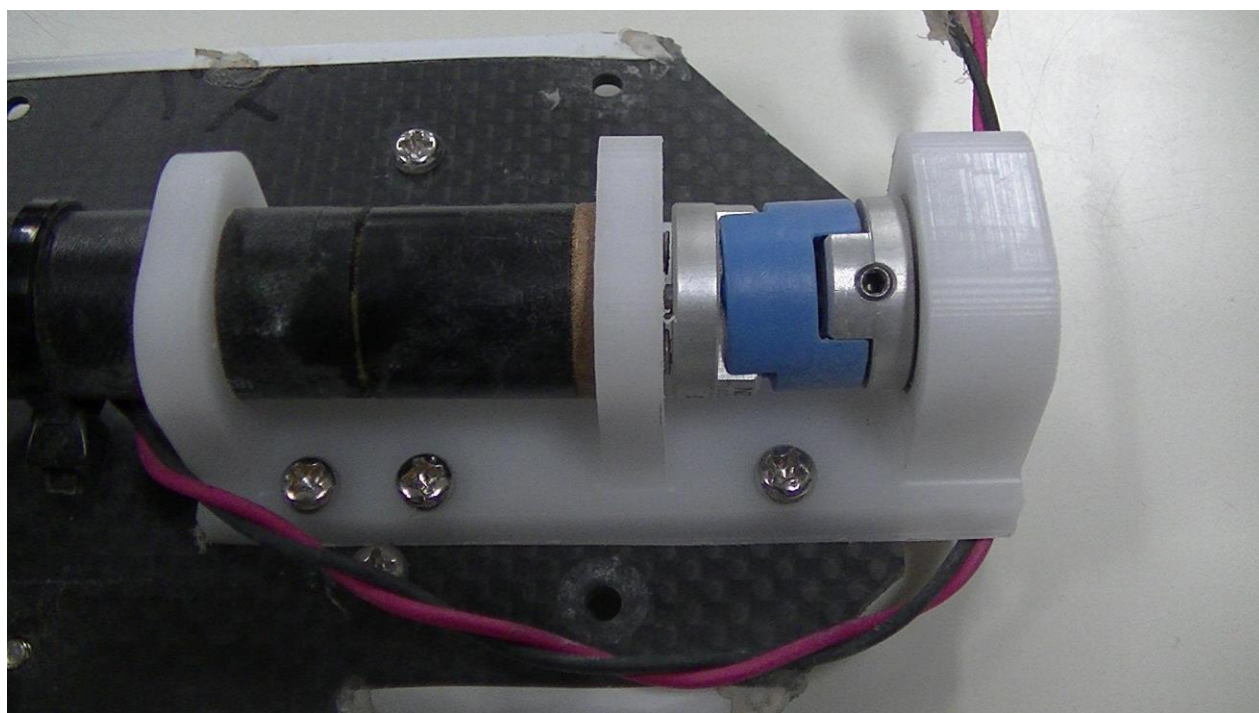


図 2.7.11: カップリング

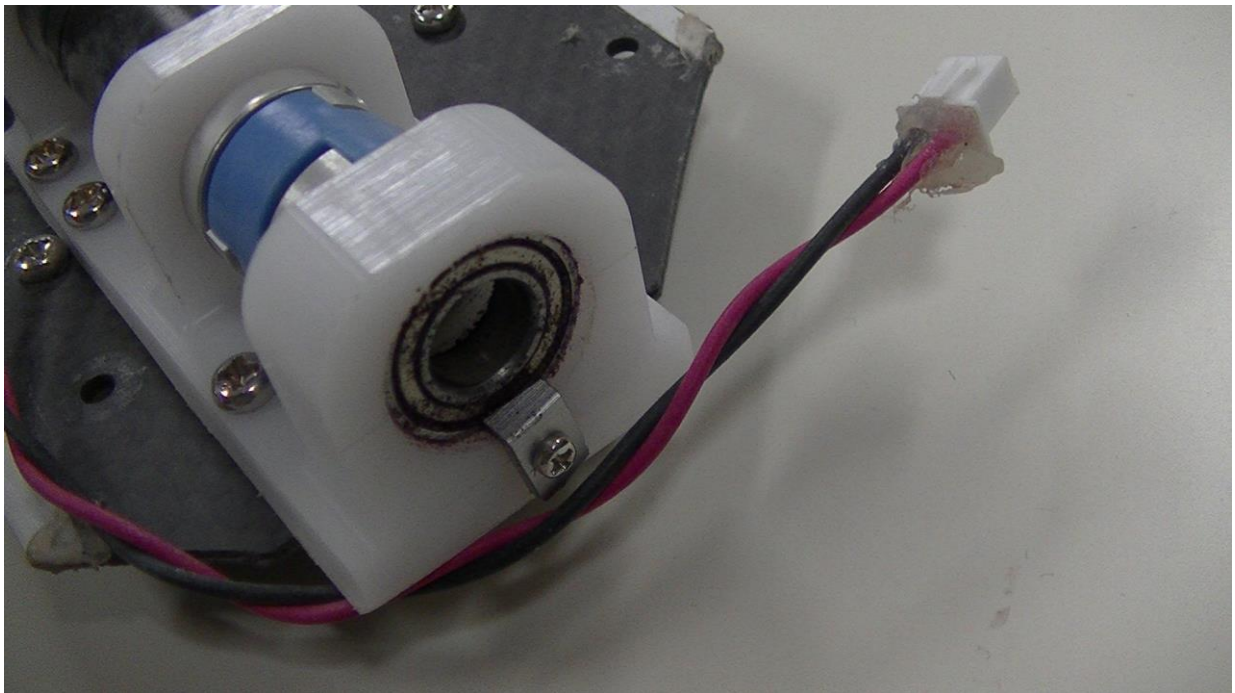


図 2.7.12: ベアリング

- 回路設計

図 2.7.13, 14 に写真を示す。実装はユニバーサル基板上で行った。質量，体積などの物理的な制約上，電源は単一の電池(リチウム乾電池 6 本)をレギュレータで 3.3V, 5V に降圧し，それぞれのモジュールに供給している。ただし，モーターは 9V で駆動している。

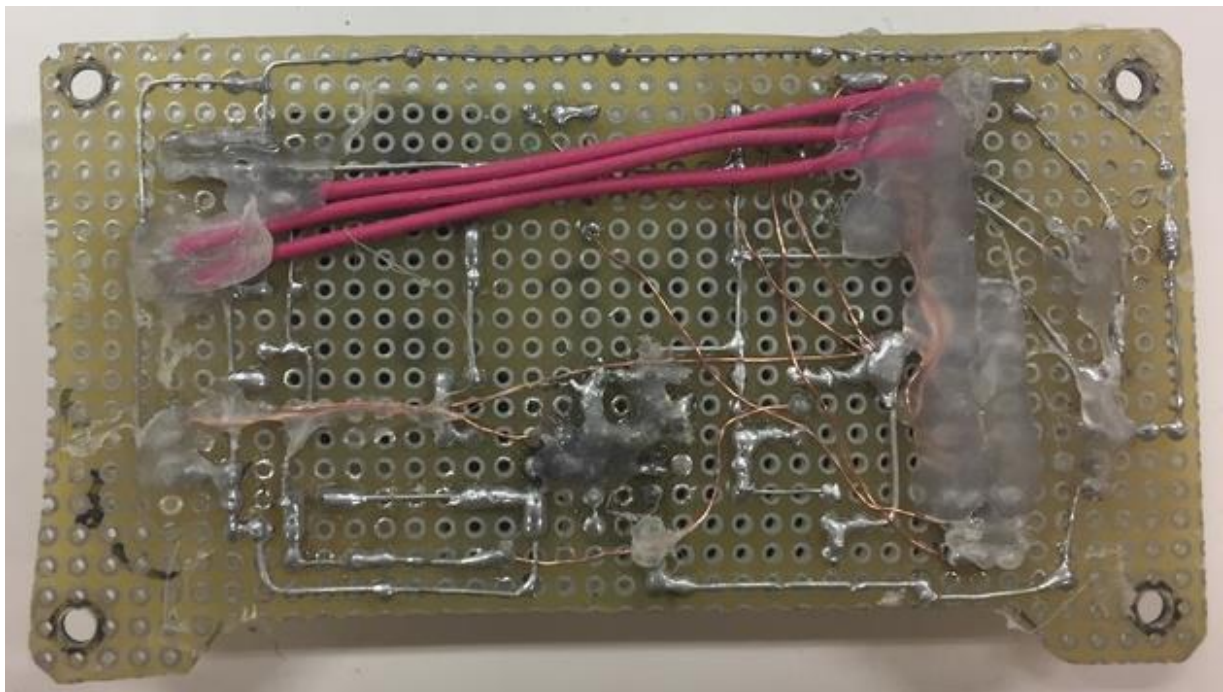


図 2.7.13: 回路裏面



図 2.7.14: 回路表面

2.8 プログラム・アルゴリズム

使用言語: スマートフォンは Java 及び XML(Android プログラミング), マイコンは Arduino 言語

【メインプログラム】

図 2.8.1 にフローチャートを示した.

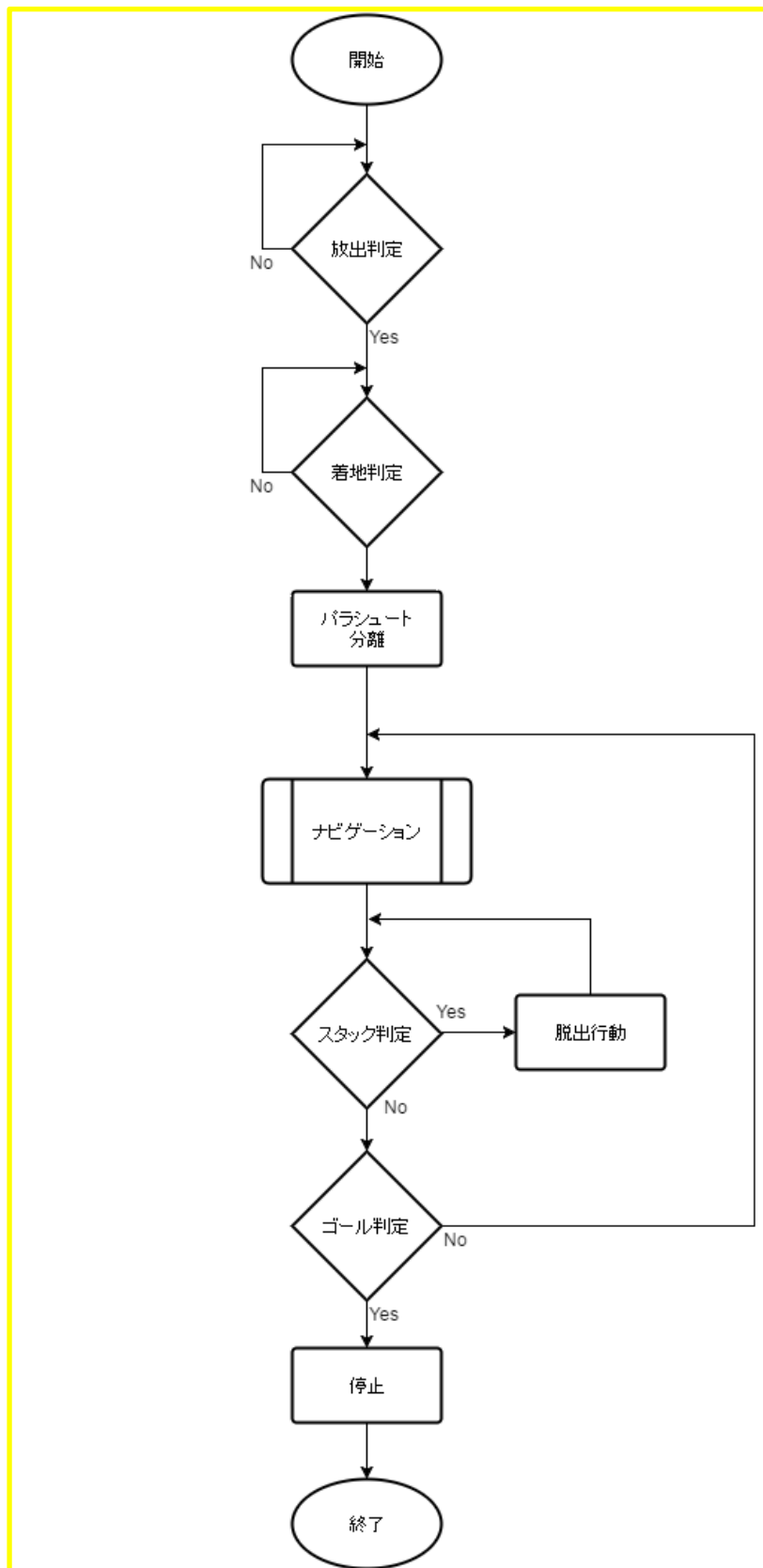


図 2.8.1: フローチャート

オールシーケンスのフェーズはスタンバイ・ランディング・セパレーション・ナビゲーション・エスケープ・ゴールの6つのシーケンスに分かれている。以下に詳細を示す。

1. スタンバイシーケンス

オールシーケンスが始まるとまずこのシーケンスになる。スタンバイシーケンスはキャリアに格納され、ロケットが発射し、CanSatが放出されるまでの間のシーケンスである。光センサーの値を監視し、その値が大きくなった時をキャリアからの放出と見なしランディングシーケンスに移行する。



図 2.8.2: スタンバイシーケンスのスマホの画面

2. ランディングシーケンス

キャリアからの放出判定後から地面に着陸するまでの間のシーケンスである。加速度センサー及び気圧センサーの値を監視し、その変位が十分に小さい時間が十分に続いたときに着地判定し、セパレーションシーケンスに移行する。



図 2.8.3: ランディングシーケンスのスマホの画面

3. セパレーションシーケンス

パラシュートを切り離すシーケンスである。着地後、回路との通信を確立し、サーボモーターを動かすことで機体に取り付けられたパラシュートのピンを外し、パラシュートを分離する。一定回数サーボモーターを動かすと、ナビゲーションシーケンスに移行する。



図 2.8.4: セパレーションシーケンスのスマホの画面

4. ナビゲーションシーケンス

ゴール地点へのナビゲーションを行うシーケンスである。センサーとして用いるのは GPS のみである。GPS の座標を取得できていないかもしくはその精度が悪い時には高精度な座標を取得できるまで徐行を行う。ゴール地点の座標と現在地の座標を用いて Android 標準ライブラリにより、北を 0 度としたときの現在地からゴール地点へ方向(角度)と距離を求め、GPS から得られる現在向いている方向(角度)からその差分を計算する。差分が小さい場合は直進、大きい場合は右左折、極めて大きい(正反対)の場合は旋回する。直進の場合は両モーターの出力を最大にし、右左折は片方のモーターの出力を最大に、もう片方の出力を 0 にすることで行い、旋回の場合はその時間を増やす。なお、一定時間内での GPS による移動距離が小さい時はスタックもしくは横転とみなし、エスケープシーケンスに移行する。ゴール地点への距離が極めて小さい時、ゴール判定し、そのシーケンスに移行する。



図 2.8.5: ナビゲーションシーケンスのスマホの画面

5. エスケープシーケンス

ナビゲーションシーケンスにおいて、GPS による一定時間内の移動変位が少ない間、このシーケンスが作動する。移動変位が大きくなるとナビゲーションシーケンスに戻る。このシーケンスでは轍にスタックもしくは横転からの復帰を目的として、脱出を試みる。そのため、(1)両モーター出力最大で轍を乗り越える (2)後退し障害物から離れる (3)ランダムに前後に動き轍から抜け出す という動きを繰り返す。

6. ゴール

ナビゲーションシーケンスにおいて、ゴールと現在地の距離が一定以内の時にこのシーケンスに移行する。このシーケンスに入ると CatSat は停止し、ゴール判定をする。また、各種ログの書き出しや統計データの集計、通ったルート(パス)を記録し、プログラムが終了可能な状態になる。



図 2.8.6: ゴール時のスマホの画面

センサー値の取得は Android API を用いる。GPS 情報、気圧センサー、照度センサーは 1 秒に 1 回、加速度センサーは 0.25 秒に 1 回値を取得し、加速度センサーに関しては 3 回分の平均値をプログラム内で使用している。

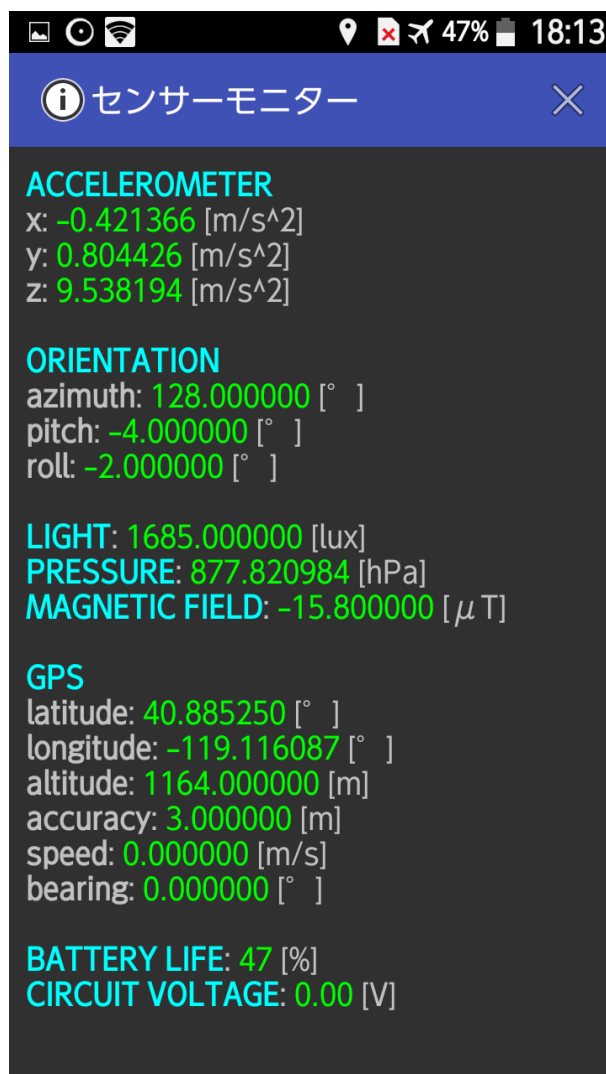
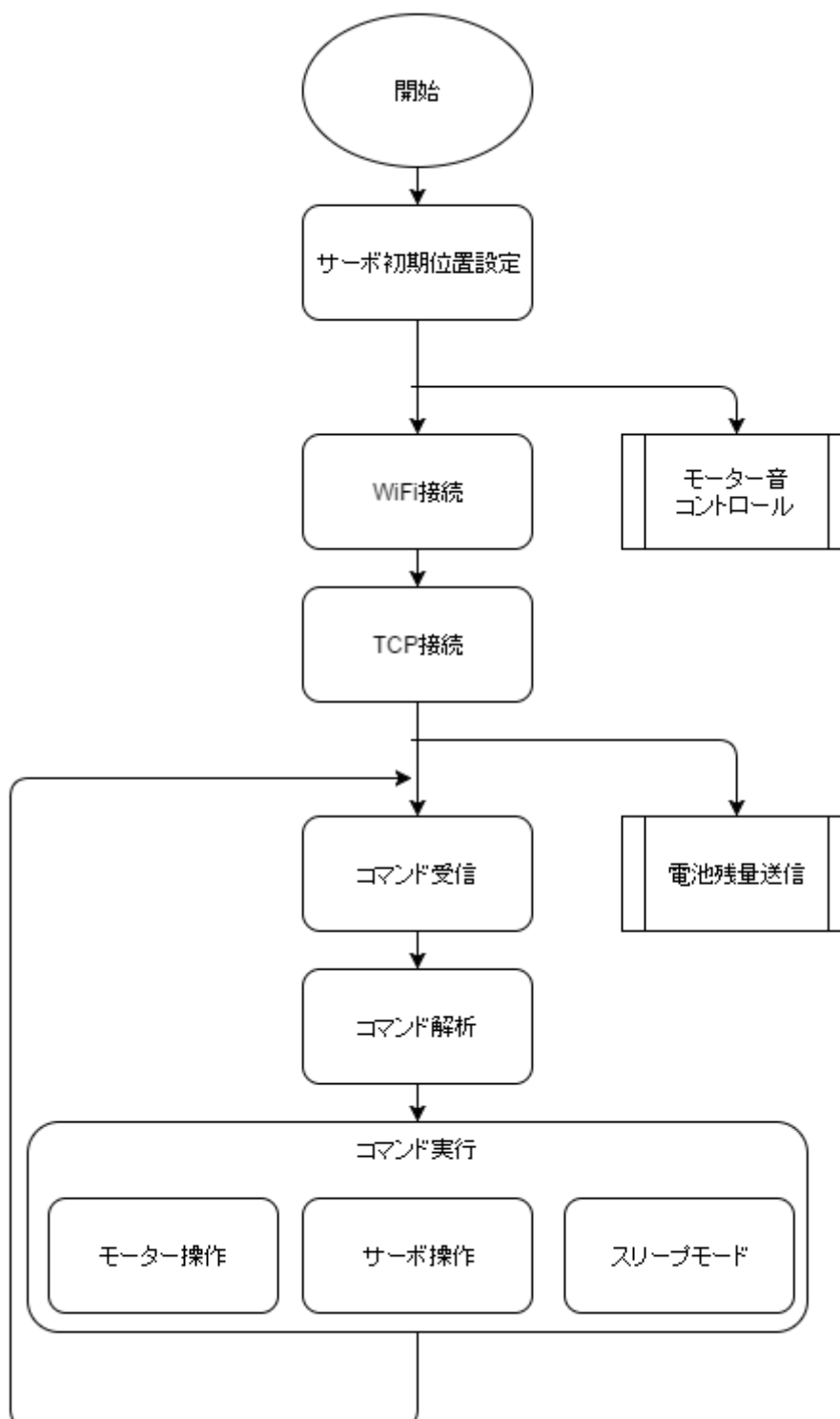


図 2.8.7: センサー値を表示させている画面

【通信関連】

通信には IEEE 802.11 規格を用い、通信プロトコルに Transmission Control Protocol を用いた。スマートフォンからマイコンにモーターを動かすなどの指令を通信する。スマートフォンがサーバー、マイコンがクライアントの構成とした。この通信はシーケンス中は維持され、通信の切断が起きた際には復帰処理が行われる。また、通信の制御として、コマンド別に 100 ミリ秒間隔の通信制限を設けた。

【マイコン】



ハードウェアはマイコン内蔵 Wi-Fi モジュール”ESP-WROOM-02”(Expressif 社)を用いた。利点として、Arduino コンパチブルのファームウェアがあるため、開発が容易であり、システムをコンパクトに纏められることが挙げられる。

プログラム実行後、サーボモータを初期位置(キャリア搭載状態)に動かし、スマートフォンとの Wi-Fi 接続、そして TCP サーバーへの接続を行う。マイコンが文字列を受信した場合の命令を解釈し、実行を繰り返す。TCP 接続がされている場合、周期的に現在の電池の電圧を送信する。着地衝撃により、マイコンが異常終了した場合、通信の復帰が困難になるため、着地時はスリープモードにし、Wi-Fi を切断した状態にする。

また、通信とは独立して、モータードライバへの信号の PWM 周期を変化させることで、モーターの磁励音に音階をもたせ、音楽を奏でる。

2.9 特に工夫した点・苦労した点

【工夫した点】

- フカフカの轍にはまらないようにするため、履帯を用いた一輪クローラー型にした
 - ロッドと呼ばれる木の棒により、履帯がたわまずに走破性が増した
- スマートフォンを乗せながらも重心がなるべく安定するような配置とした
- 着地衝撃によって壊れない強固な機構にした
- 昨年、電池 BOX がずれることによって生じる問題があったため、電池 BOX を中に入れ、周りに影響を与えないようにした
- スタビライザーを可動式にし、適用性を広げた
 - 履帯の張力の調節
 - 接地面積の増加
 - 轍にはまった際の脱出動作
- スマートフォンは高価なため、回路にショートなどの異常があったときにそれによってスマートフォンが壊れるようなことがないように、回路とスマートフォンは無線による通信を行うことにした
- CanSat が現在の状態を喋ることで自ら実況を行う。これによりステータスがわかりやすくなった
- カプセル化を意識してプログラムが設計されているため、仕様変更などに強い
- スマートフォンから全ての操作が可能のため、各種実験時に PC を持ち運ぶ必要がない
- モーターの磁励音の有無で、ローバーが制御下にあるか判断が可能

【苦労した点】

- 一輪のため、方向制御のアルゴリズムを考案するのに苦労した
- スマートフォンの体積が大きいため、基礎設計の際に苦労した
- 履帯の素材や形状・構造を決定するために何度も試作・実験を繰り返した
- 履帯から中の駆動部が抜けださないようにするために様々な実験を行った
- 履帯の重量が重たいため、軽量化にも苦労した
- マイコンの耐衝撃対策
- 履帯に囲まれた機体からパラシュートを放出しなければならないこと
- 機体の重量の減量

3 大会結果

3.1 能代宇宙イベント

3.1.1 目的

- 一輪クローラー型 CanSat の草地走行の性能の調査
- 高所からのパラシュートの開傘と着地衝撃による機体への影響の調査
- チーム団結力と技術力の向上
- ARLISS に向けた予行練習

能代宇宙イベントにおけるサクセスクライテリアは次のとおりである。

表 3.1.1.1: 能代宇宙イベントでのサクセスクライテリア

ミニマムサクセス	①パラシュートを切り離すことができる。 ②100m 以上走行することができる。
ミドルサクセス	③方向転換において要求された角度に対する誤差を 15 度以内に抑えることができる。
フルサクセス	④二輪ローバーがスタックする轍や叢にスタックせずに走行できる。 ⑤ゴール地点から 10m 以内で停止する。
アドバンスドサクセス	⑥CanSat が通常走行で越えられない轍や叢によって進行不可能な時に脱出機構及び脱出プログラムによって脱出できる。

3.1.2 結果

サクセスクライテリアの達成度合いを次の表で示す。

表 3.1.2.1: 能代宇宙イベントでのサクセスクライテリアの達成具合

項目	競技中	イベント全体	備考
①	×	○	現地で行った実験ではパラシュートを切り離すことが確認できている。
②	×	○	現地で行った実験では 100m 以上走行している。
③	×	△	現地で行った実験では方向推定は出来ているものの、正しく左折出来ないことがあった。
④	×	○	現地で行った実験では刈られていない草地であっても走行可能であった。
⑤	×	○	現地で行った実験では成功している。
⑥	×	○	現地で行った実験ではどうしても越えられない切り株などから脱出プログラムにより回避できた。

表 3.1.1.2: 能代宇宙イベント競技の結果

能代	<u>パラシュート開傘</u>	<u>壊れずに着地</u>	<u>パラシュート分離</u>	<u>走行開始</u>	<u>10m 以上制御走行</u>	<u>50m 以上制御走行</u>	<u>ゴール</u>

1回目	○	○	×	×	×	×	×
2回目	○	×	×	×	×	×	×

投下 1 回目の動画: <https://www.youtube.com/watch?v=sGRJt7eN4Io>

投下 2 回目の動画: <https://www.youtube.com/watch?v=QKfsi6b-zzY>



図 3.1.2.1: 競技の様子

(左から 放出, 展開, 着地, パラシュート切り離せずナビゲーション開始)

競技 1 回目は, パラシュートの開傘・着地には成功したがパラシュートの分離に失敗した. パラシュートを切り離すサーボモーターのピンを外す力が足りず, サーボモーターの動作角度が非常に小さく, 振動していた. その後の動作確認でサーボモーターが壊れていることを確認した. (この様子は次の動画を参照: <https://www.youtube.com/watch?v=je5VkiB93jc>)

壊れたサーボモーターを交換し, 競技 2 回目に臨んだが, 制御用マイコンが暴走し, 制御が不能になり, リタイアした.

これらの原因追求や対策などは 3.1.4 に記した.

3.1.3 取得データ

2 回ともパラシュートの切り離しに失敗したため, ナビゲーションにおける制御履歴はないが, それ以外のログは残っているその抜粋をここに記す. 全てのログを載せると非常に膨大な量のため, 同ディレクトリに「ログ」というディレクトリに全てのログを添付してある.

【1 回目】

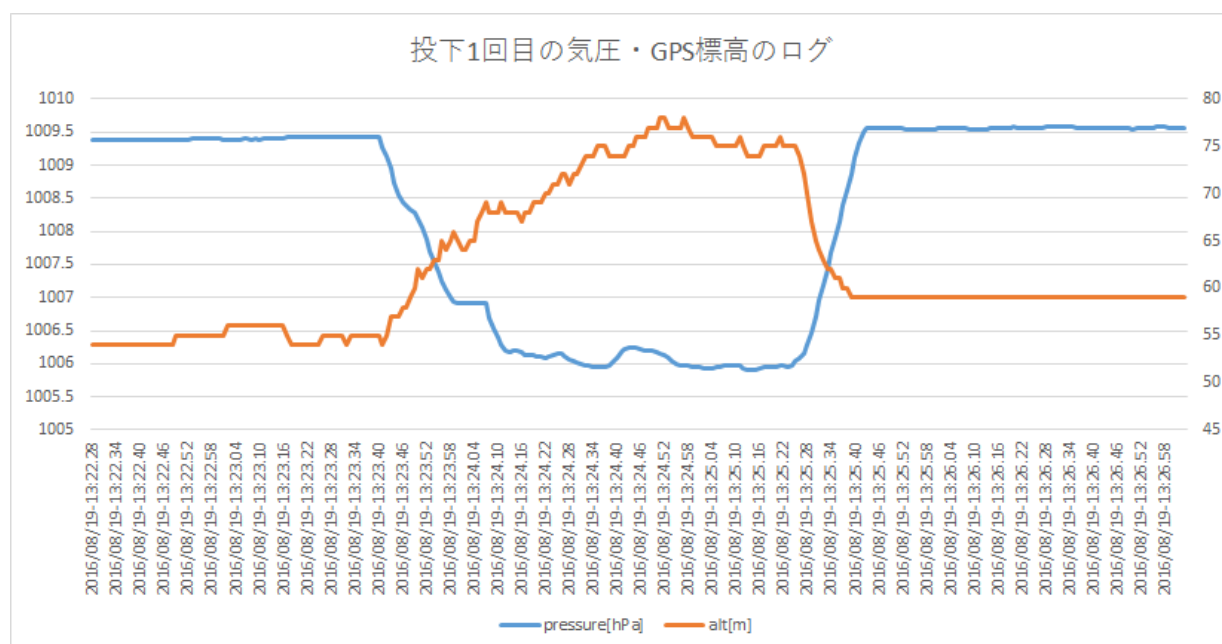


図 3.1.3.1: 投下 1 回目の気圧・GPS 標高ログ

左軸が気圧[hPa], 右軸が標高[m]である. グラフより, 気球により 20m 程上昇し, その際は気圧が下がっている.

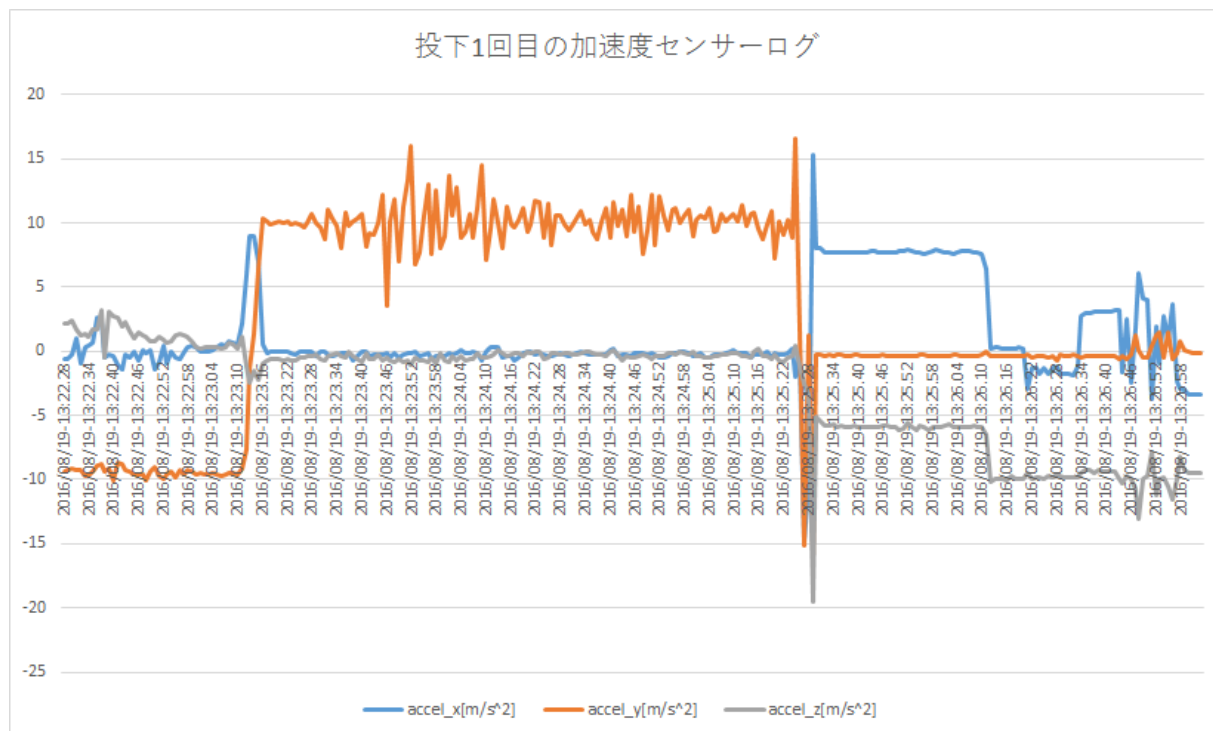


図 3.1.3.2: 投下 1 回目の加速度ログ

縦軸は加速度[m/s/s]である。

標高が小さくなった時刻に着地による大きな衝撃が加わったことがわかる。

【2 回目】

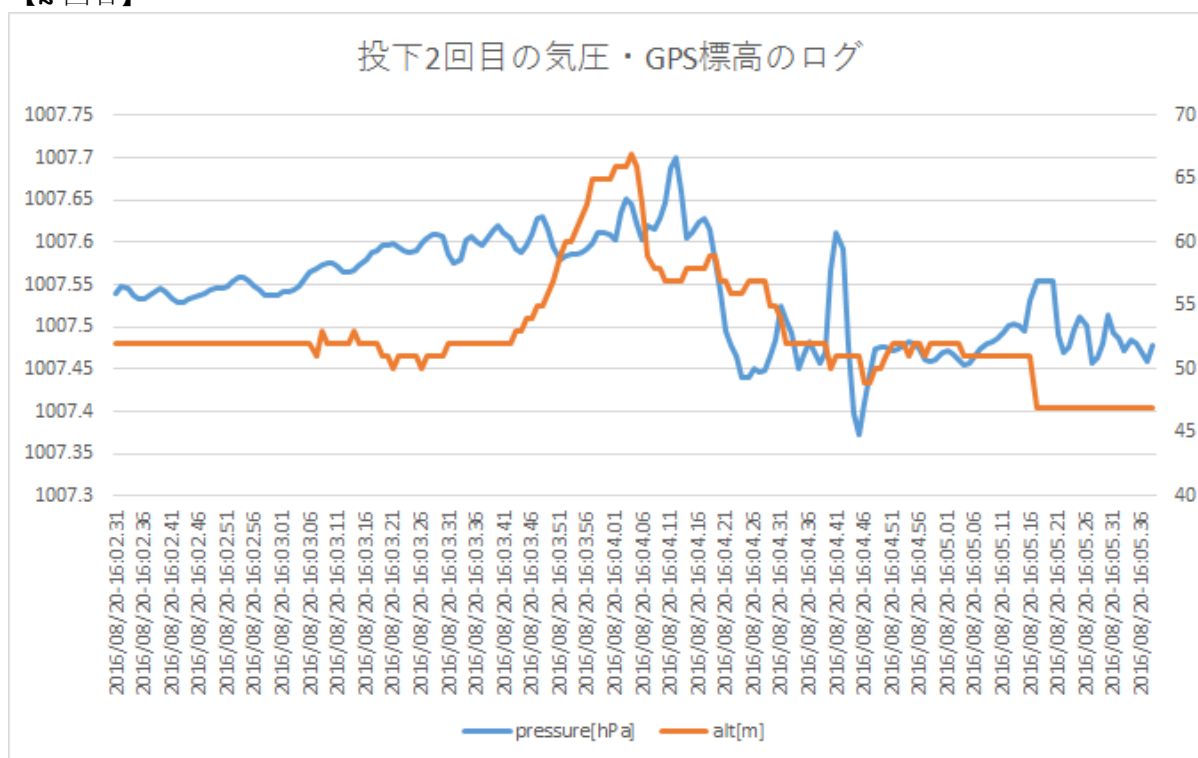


図 3.1.3.3: 投下 2 回目の気圧・GPS 標高ログ

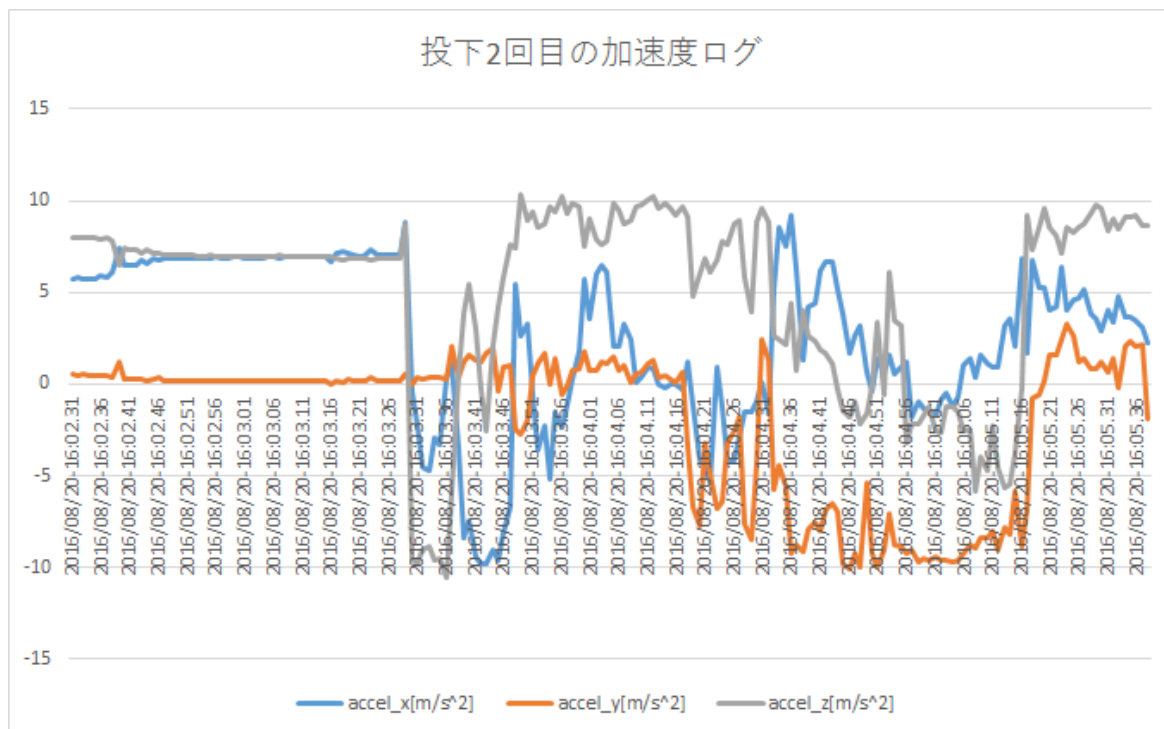


図 3.1.3.4: 投下 2 回目の加速度ログ

3.1.4 故障原因解析

【1 回目】

パラシュートのピンを抜く力が足りず、サーボモーターに力が加わり続け(過負荷)、サーボモーターが壊れてしまったことにより、パラシュートの切り離しに失敗した。また、その際、回路超電流が流れ、マイコンの一部ピンの機能を壊した。パラシュートのピンを抜く力が足りなかった原因は、マイコンスリープ時にスマートフォンが送信した生存確認命令が TCP 通信の性質上残り続け、マイコンのスリープからの復帰時に全ての命令がマイコンに届き、それにパラシュートを切り離すためのサーボモーターを動かす命令も含まれていたのだが、マイコン側が溜まっていた命令を処理しきれず、サーボモーターを動かすタイミングが短くなってしまったものによる。この問題の対策として、スリープ処理中におけるスマートフォンからの命令送信の停止を行い、解決した。また、万一過電流が流れてもマイコンが壊れないように、保護回路を組み込んだ。

【2 回目】

1 回目で故障したマイコンの修理に時間がかかってしまい、自分たちの競技開始時間に間に合わず、受付〆切間近になってしまった。その時間帯には発電機の燃料がなくなってしまう電力の供給が止まってしまっていた。そのため本来であればグルーガンでマイコンと基盤を固定するのが、それができず、また他の手段を講じている時間もなかったため、グルーガンによる固定をせずに競技に臨んだ。結果として、着地時の衝撃でマイコンのピンが基盤から若干浮いてしまい、それによってマイコンに予期しない入力が入ってしまったこと原因となり、マイコンの制御ができなくなる状態(暴走)してしまった。対策として、マイコンの固定をしっかりと行うことが大事である。

3.2 ARLISS

3.2.1 目的

作成した一輪クローラー型 CanSat の有効性を検証し、ロケット打ち上げからゴールに到達するまでの要求が満たされたものであるのかの検証を行う。

表 3.2.1.1: ARLISS サクセスクライテリア

	内容
ミニマムサクセス	①100m 以上走行することができる。
フルサクセス	②現地の走行実験において従来の二輪ローバーに比べ、50%少ないスタック回数で砂地を走行する。 ③ゴール地点から 10m 以内で停止する。
アドバンスドサクセス	④CanSat が超えられない轍によって進行不可能な時に脱出機構及び脱出プログラムによって昨年度製作した CanSat より 50%早く脱出することができる。 ⑤センサーの値により、左右のモーターのレシオを自動調整し、直進制御することができる。

3.2.2 結果

サクセスクライテリアの達成度合いを次の表で示す。

表 3.2.2.1: ARLISS でのサクセスクライテリアの達成具合

項目	結果	備考
①	○	1st flight と 2nd flight での合計走行距離は 16.2307km である。
②	○	1st flight では 1 回、2nd flight では 1 回もスタックすることはなかった。また、従来の 2 輪ローバーが超えられない轍において検証したところ、50%以上少ないスタック回数で砂地を走行可能であった。
③	○	2nd flight ではゴールから 4.80m 地点で停止した。
④	○	1st flight では全ての轍スタックから脱出した。2nd flight では 1 回もスタックしたなかった。
⑤	△	左右のレシオ差は履帯が吸収するため、この機能は最終的に実装する必要がなかったため、未実装のまま本番を迎えた。

表 3.2.2.2: ARLISS 競技の結果

	<u>パラシュート開傘</u>	<u>破損なしでナビゲーション開始</u>	<u>パラシュート分離</u>	<u>走行開始</u>	<u>10m 以上制御走行</u>	<u>100m 以上制御走行</u>	<u>1km 以上制御走行</u>	<u>ゴール</u>
1 回目	○	○	○	○	○	○	○	×
2 回目	○	△	○	○	○	○	○	○

2 回目ではナビゲーション開始時にサーボモーターの動きが鈍くなったため、「破損無しでナビゲーション開始」を△としたが、それによる走行への影響は無い(ナビゲーション以降に使用していないサーボモーターだったため)。

1 回目のフライト動画：<https://youtu.be/anIcrxIKMZQ>

2 回目のフライト動画：<https://youtu.be/BKAEhwr2DtA>

3.2.2.1 一回目

着地し、deep sleep が解除されるのを待機している。



図 3.2.2.1: 着陸時

deep sleep が解除されると、通信を復帰しパラシュートを切り離すことに成功した。



図 3.2.2.2: パラシュート切り離し時

轍にはまりスタックした。ランダム行動によってスタック判定から 10 分で抜け出すことが出来た。



図 3.2.2.3: スタック時

19:14 分頃、ゴールまで残り 750m 付近で CanSat が右折判定のみを繰り返すようになってしまった。

1 時間ほど粘り、時折左折の判定もしていたが右折判定を繰り返していたためリタイアした。直後にスマートフォンのセンサー値を確かめてみると、正しい方角を取得できていなかった。

3.2.2.2 二回目

着地し、deep sleep が解除されるのを待機している。



図 3.2.2.4: 着陸時

deep sleep が解除されると、通信を復帰しパラシュートを切り離すことに成功した。走行を始めるがスタビサーボが展開されていなかった。しかし、徐々に正常な角度まで展開した。ゴール後に、スタビサーボは壊れていたことが確認された。



図 3.2.2.5: パラシュート切り離し時



図 3.2.2.6: ゴール時

3.2.3 取得データ

全てのログを載せると非常に膨大な量のため、同ディレクトリに「ログ」というディレクトリに全てのログを添付してある。ここではその概要を示す。

3.2.3.1 一回目

- ゴール地点までの距離: 783m
- 走行距離: 14108m
- 走行時間: 3 時間 22 分
- 制御開始: 2016/09/13-17:02.13, 座標 (40.8852498,-119.1161018), ターゲットまでの距離 3959.6882m

- 制御終了: 2016/09/13-20:24.20, 座標 (40.88481008 -119.12647232), ターゲットまでの距離 875.403m

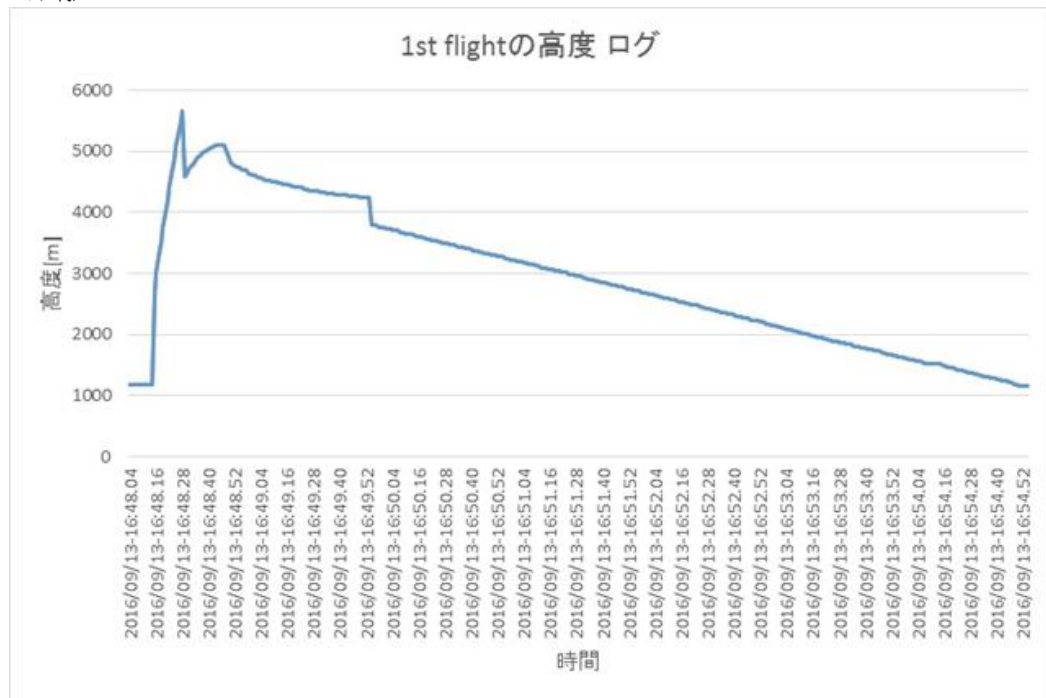


図 3.2.3.1: 1st flight の高度ログデータ

ロケットにより 5000m 以上の高度まで到達し、その後落下していることがわかる。

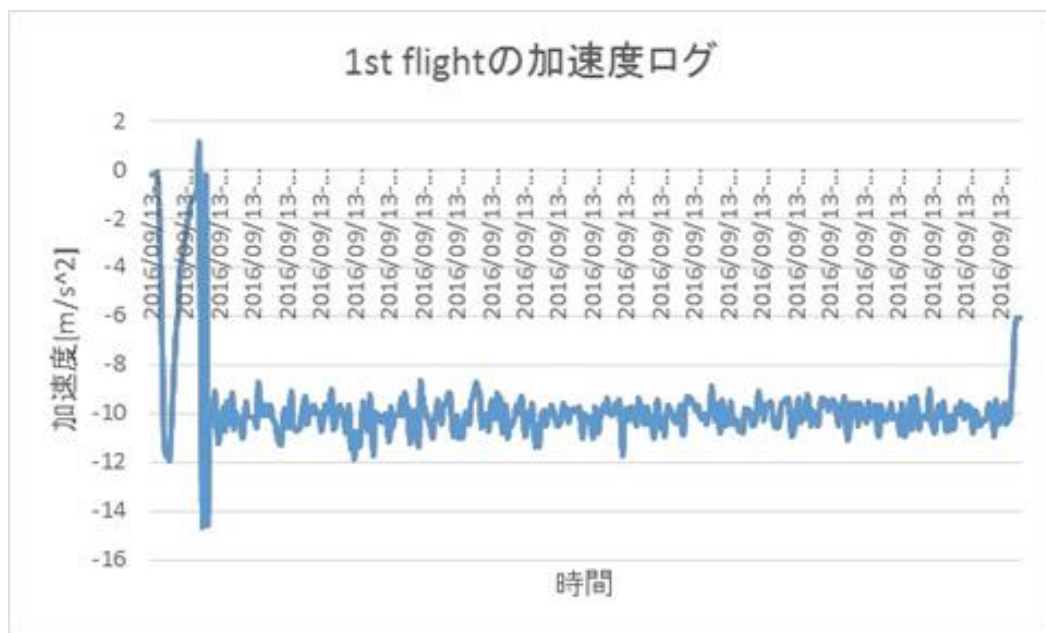


図 3.2.3.2: 1st flight の加速度ログデータ

打ち上げの瞬間と着地の瞬間に大きな加速度が加わっていることがわかる。

- 16:47 打ち上げ
- 16:48 最高高度に到達(高度 5669m), 放出
- 16:54 着地(高度 1163m)
- 17:01 パラシュート切り離し
- 17:02 ナビゲーション開始(ゴールまで 3959.7m)
- 17:50 スタック
- 20:24 リタイア(走行距離:14.1km, 走行時間:3 時間 22 分)

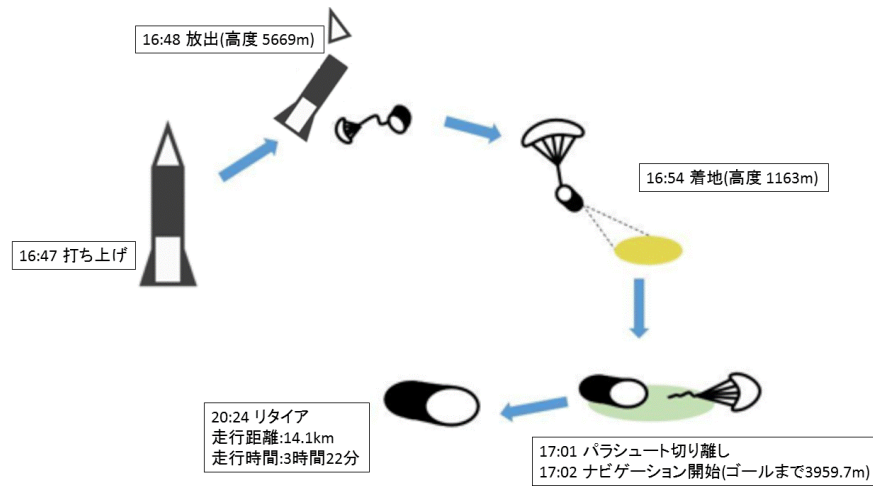


図 3.2.3.3: 1st flight 取得データ時間



図 3.2.3.4: 1st flight 走行経路

3.2.3.2 二回目

- ゴール地点までの距離: 4.80m
- 走行距離: 2112.7m
- 走行時間: 23 分 51 秒
- シーケンス開始からゴールまでの時間: 1 時間 6 分 16 秒
- 制御開始: 2016/09/15-14:45.20, 座標 40.8852498, -119.1161018, ターゲットまでの距離 1855.0623m
- 制御終了: 2016/09/15-15:08.48, 座標 40.88523865 -119.11610781, ターゲットまでの距離 1.337825m

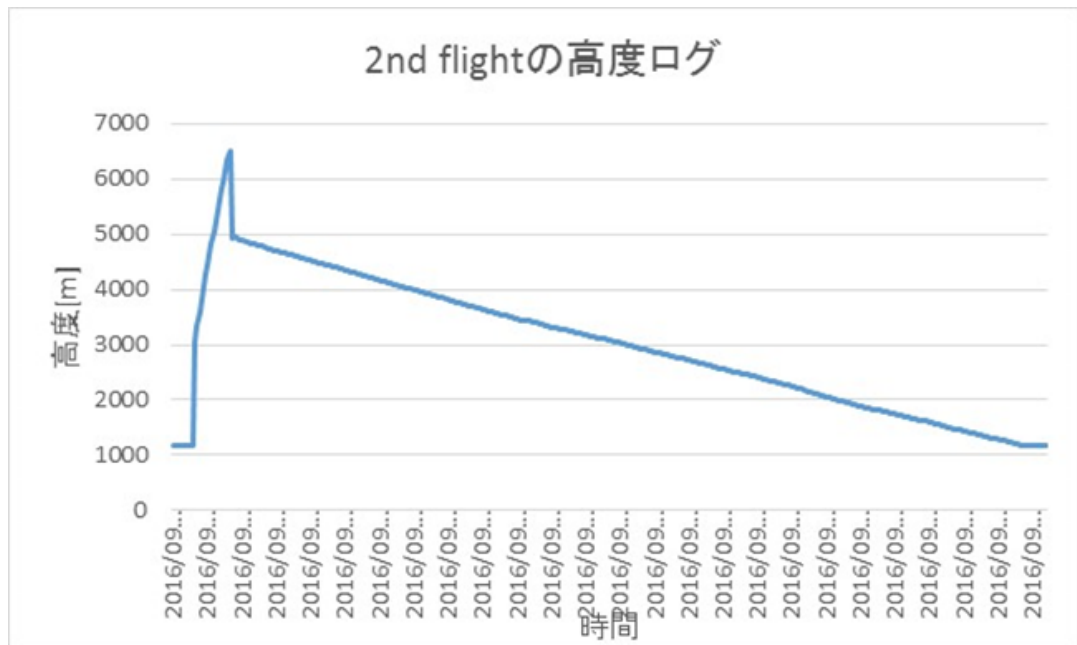


図 3.2.3.5: 2nd flight の高度ログデータ

ロケットにより 6000m 以上の高度まで到達し，その後落下していることがわかる。

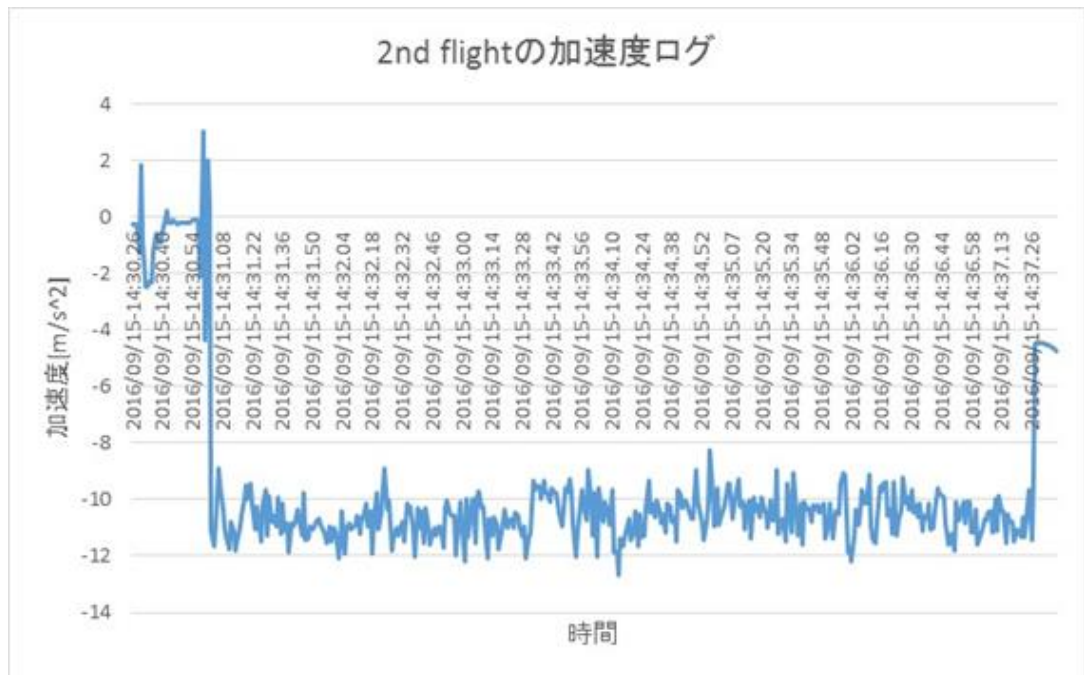


図 3.2.3.6: 2nd flight の加速度ログデータ

打ち上げの瞬間と着地の瞬間に大きな加速度が加わっていることがわかる。

- 14:30 打ち上げ
- 14:30 最高高度に到達(高度 6522m),放出
- 14:37 着地(高度 1174m)
- 14:44 パラシュート切り離し
- 14:44 ナビゲーション開始(ゴールまで 1855.1m)
- 15:08 ゴール(走行距離:2112.7, 走行時間 23 分 51 秒)

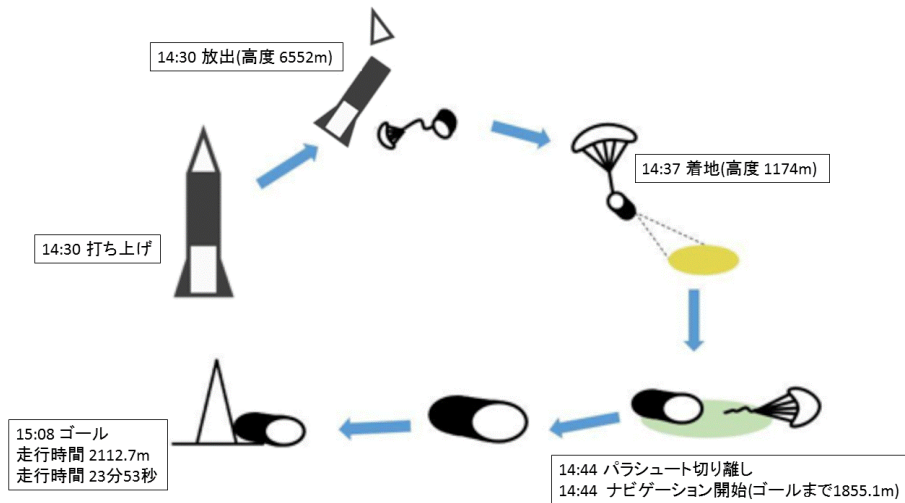


図 3.2.3.7: 2nd flight 取得データ時間



図 3.2.3.8: 2nd flight 走行経路

3.2.4 故障原因解析

1st flight では以下の 2 つの問題点があった。

- 左旋回性能が低い
 - ハードウェアの不良
 - 直進命令実行時，右寄りに進む履帯を使用していた
 - 2nd flight では履帯の張り方を改善
 - 左旋回命令の実行がされないことがある
 - 一部コマンドが捨てられていた
 - マイコンが通信により送られてきた命令信号を処理しきれない時にその命令を捨てることがある。左折コマンドは命令が右折コマンドに比べてコマンド数が多いため，左折コマンドが捨てられる可能性が高く，左旋回性能の低下につながった。
 - 2nd flight ではコマンドを簡素化し，処理が飽和されないよう対策を施した。
 - ログデータおよび走行経路より左折の展開性能が向上していることがわかる。このため，この対策の有効性が確認できた。
- 走行開始から 2 時間半経過したあたりから正常なナビゲーションができていない

- 地磁気センサーによる方向推定が誤っていた
 - 地磁気センサーは時間経過によりキャリブレーションの効力がなくなってしまうことが原因
 - 2nd Flight では方向推定に GPS を用いることによってこの問題を解決した。

4 まとめ

今回のクローラー型 CanSat の開発と ARLISS での実験を経て、クローラー型 CanSat の走破性が非常に高く、悪路に有効なことが証明された。